Traffic speed prediction using ensemble kalman filter and differential evolution

Lukáš Rapant

IT4Innovations, VŠB – Technical University of Ostrava, Ostrava-Poruba, Czech Republic

Abstract. Importance of traffic state prediction steadily increases with growing volume of traffic. Ability to predict traffic speed in short to medium horizon (i.e. up to one hour) is one of the main tasks of every newly developed Intelligent Transportation System. There are two possible approaches to this prediction. The first is to utilize physical properties of the traffic flow to construct an exact or approximate numerical model. This approach is, however, almost impossible to implement on a larger scale given the difficulty to obtain enough traffic data to describe the starting and boundary conditions of the model. The other option is to use historical traffic data and relate information and patterns they contain to the current traffic state by application of some form of statistical or machine learning approach. We propose to use combination of Ensemble Kalman filter and Cell Transmission Model for this task. These models combine properties of physical model with ability to incorporate uncertainty of the traffic data.

1 Introduction

The objective of this study is to propose a macroscopic traffic speed prediction model, which can be based both on stationary and dynamic data sources. Most straightforward solution to the traffic speed prediction problem is to solve it by application of some form of kinematic wave model or car following model. They accurately describe behavior of traffic flow by application of physical models. Examples of this approach are articles written by Tang et. al. [1] and [2] and Costesqua et. al. [3]. However, the absence of traffic intensity or density information in FCD and sparsity of other sensors rules out most of these traffic flow models, which are based these quantities. Even though there are some models which circumvent this problem (for example Cell transmission model - velocity (CTM-v), I still have insufficient data to exactly describe their initial and boundary conditions. While their results may seem realistic (as a dissolution of traffic jam from my CTM-v model shown in the Figure 1, their quality is quickly eroding with each iteration. Therefore, other prediction scheme for this problem must be proposed.

Other possible approach to traffic state prediction is to use historical data about traffic and relate them to the current state of the traffic by application of some statistical or machine learning approach. This can be described as a task of finding traffic state time series prediction

$$P_l = \max_{s \in S} p(s \mid x_1, \dots, x_n),$$

where P_l is prediction of length l, S is set of possible predictions of certain length l, s is member of S and $x_1, \dots, x_n$ are n last measurement of speed.

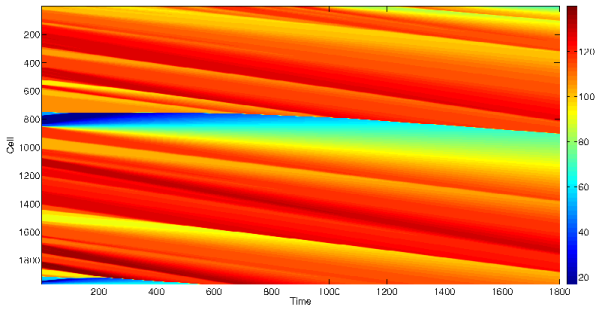

Figure 1. Result from simulation of D1 by CTM-v model (time is in seconds and color represents speed in km/h)

Examples of this approach are described by Calvert et. al. [4] or Huang [5] and some other authors. This task is, however, also not a simple one. It is complicated by complex nature of the traffic data. They can, depending on type of their source, be inaccurate or not available in required quantity. Despite this fact, this approach is preferable in case of traffic state prediction on Czech Republic highways due to the unavailability of boundary condition data (i.e. there are not measurements on all ramps leading to and out of motorway).

One of the possible approaches is Ensemble Kalman filter (EnKF). EnKF model was, amongst other reasons, chosen because its ability to work with coarse physical model and to represent nonlinearity of the traffic. For the physical model, CTM-v was chosen because it is only physical model that requires only speed for the

computation (and FCD contain only information about speed, not density or intensity). However, its application would require finding heuristics for boundary conditions (beginning and end of the modelled road and ramps). This heuristic will be based on analysis of historic data. Then this model will be implemented into Ensemble Kalman filter and resulting model will be tested and optimized on the historical data. EnKF is an algorithm that uses a series of measurements observed over time, containing noise and other uncertainties. It produces estimates of unknown variables that tend to be more precise than those based on a single measurement alone.

2 Theoretical background

2.1 Kalman filter

The Kalman filter is used to solve the general problem of estimating the state of a discrete-time controlled process that is defined by the linear stochastic difference equation

$$x_k = Ax_{k-1} + Bu_k + w_{k-1}, \quad (1)$$

with a measurement $z \in R^m$ that is

$$z_k = Hx_k + v_k. \quad (2)$$

The matrix A in the difference Equation 1 relates the state at the previous time step to the state at the current step. The matrix B relates the optional control input to the state x. The matrix H in the Equation 1 relates the state to the measurement z_k. The random variables w and v represent the process and measurement noise. They are assumed to be independent and with normal probability distributions.

$$P(w) \sim N(0, Q) \quad (3)$$
$$P(v) \sim N(0, R) \quad (4)$$

Q and R are the process noise covariance, respectively measurement noise covariance matrices.

The Kalman filter estimates a process by using a form of feedback control: the filter estimates the process state at some time and then obtains feedback in the form of (noisy) measurements. As such, the equations for the Kalman filter fall into two groups: time update equations and measurement update equations. The time update equations are responsible for advancing the current state and error covariance estimates in time to obtain the a priori estimates for the next time step. The measurement update equations are then responsible for the feedback. Their task is to incorporate a new measurement into the a priori estimate to obtain an improved a posteriori estimate. The time update equations can be considered as prediction equations, while the measurement update equations can be considered as correction equations.

The specific equations for the time updates are following:

$$\widehat{x_k^-} = A\widehat{x_{k-1}} + Bu_k$$
$$P_k^- = AP_{k-1}A^T + Q$$

These time update equations project the state and covariance estimates forward from time step (k-1) to step k. A is from Equation 1 and Q is from Equation 3. The specific equations for the measurement updates are:

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1},$$
$$\widehat{x_k} = \widehat{x_k^-} + K_k(z_k - H\widehat{x_k^-}),$$
$$P_k = (I - K_k H)P_k^-.$$

H is from Equation 2 and R is from equation 4. The first task during the measurement update is to compute the Kalman gain K_k. The measurement is incorporated after this step to obtain z_k, and then to generate a posteriori state estimate. The final step is to obtain a posteriori error covariance estimate.

After each time and measurement update cycle, the process is repeated with the previous a posteriori estimates used to predict the new a priori estimates. This recursive nature is one of the very appealing features of the Kalman filter because it makes practical implementations much more feasible.

2.1.1 Ensemble kalman filter

The ensemble KF (EnKF) was introduced as an alternative to the extended KF. Extended KF performance is poor when the state transition function is highly nonlinear. The EnKF belongs to a group of suboptimal estimators that use Monte Carlo or ensemble integration. The EnKF uses a collection of state vectors (called the ensemble members of system states) to propagate the state in time and to compute the mean and covariance needed for the correction step. The covariance estimated in this way is used to compute the Kalman gain. The correction step equation stays the same as in the traditional KF. The EnKF algorithm can be described by the following steps:

1. Generate N ensemble members of system states x_0^n by drawing N samples from a Gaussian distribution ($n = 1,2, \dots, N$).

2. Make a time update $\widehat{x_t^n} = A(\widehat{x_{t-1}^n})$.

3. Compute the mean of the ensemble:

$$\mu_t = \frac{1}{N}\sum_{n=1}^{N} \widehat{x_t^n}$$

4. Compute the covariance of the predicted state:

$$P_t = \frac{1}{N-1} E_t (E_t)^T,$$

where $E_t = \left[\widehat{x_t^1} - \mu_t, \ldots, \widehat{x_t^N} - \mu_t\right]$.

5. Calculate the Kalman gain:

$$K_k = P_t H^T (H P_t H^T + R)^{-1}$$

6. Obtain a new ensemble from the measurements:

$$x_t^n = \widehat{x_t^n} + K_t\left[z_t - H\widehat{x_t^n}\right]$$

7. Return to step 2.

2.2 Differential evolution

Differential evolution has proven to be an efficient method for optimizing real-valued functions. DE is trying to breed as good population of individuals as possible in loops called generations [6].

The very first step is defining constants affecting behavior of evolution algorithm. Those constants are crossover probability (CR) which influences probability of choosing a parameter of the mutated individual instead of the original one, mutation constant (F) which determines volume of mutation of the mutated individual, population size (NP) and finally trial specimen. Now I have a specimen and I can create initial population vector X.

When I have initial population prepared (called generation), loop begins. In each generation cycle, nested loop is executed. This loop is called evolution cycle. Each individual from current generation is bred to have better characteristic using mutations in evolution cycle. Algorithm finishes either if a solution is good enough (individual) or after defined number of generation loops had been executed.

For my purposes, the DE/rand/1 algorithm mutation is the best choice. The notation DE/rand/1 specifies that the vector v to be perturbed is randomly chosen and the perturbation consists of one weighted vector.

$$\mathrm{v} = \mathrm{x}_{\mathrm{r1,j}}^{\mathrm{G}} + \mathrm{F} \cdot \left(\mathrm{x}_{\mathrm{r2,j}}^{\mathrm{G}} - \mathrm{x}_{\mathrm{r3,j}}^{\mathrm{G}}\right),$$

where G is number of generation. Due to this mutation, new individuals are not affected by the temporary best individual from generation and space of possible solutions is searched through uniformly. More detailed description of Differential Evolution can be found in [6]. Differential evolution can be described by following pseudocode.

DE{X,f_cost,NP,F,CR,Specimen}

1. X: initial population (vector)
2. f_cost: function returning fitness of current solution
3. NP: population size
4. F: mutational constant
5. CR: Crossover threshold
6. Specimen: trial specimen
7. For I < Generation do begin
8. For J < NP do begin
9. Select J individual
10. Select three random individuals from population
11. Compute the new individual fitness
12. Choose a better one from both individuals to new population
13. EndFor
14. EndFor

2.3 Cell transmission model – velocity

CTM-v is based upon classical Lighthill-Whitham-Richards (LWR) partial differential equation. LWR PDE for modelling traffic on highways is:

$$\frac{\partial \rho}{\partial t} + \frac{\partial q}{\partial x} = 0 \tag{5}$$

where $\mathrm{q}(\mathrm{x},\mathrm{t})$ is flow and $\rho(\mathrm{x},\mathrm{t})$ is density of the vehicles at location x and time t. This equation is derived from hydrodynamics theory and expresses the conservation of mass for a fluid of density ρ and of flux q. Empirical relation called fundamental diagram $\mathrm{q}(\mathrm{x},\mathrm{t}) = \mathrm{Q}\big(\rho(\mathrm{x},\mathrm{t})\big)$ is used for the expression of q as a function of ρ. Q is flux function and is independent of variables. To transform LWR PDE to velocity LWR (LWR-v) PDE, I will use the specific flux function called Greenshields:

$$\mathrm{v} = \mathrm{v}_{\max}\left(1 - \frac{\rho}{\rho_{\max}}\right)$$

where $\mathrm{v}_{\max}$ and $\rho_{\max}$ denote the maximal velocity respectively the maximal density allowed by the model. This function expresses relation between density and velocity. Greenshields flux function can be easily inverted to express ρ as a function of v. Now I can substitute this expression into Equation 5 to obtain LWR-v PDE:

$$\frac{\partial \mathrm{v}}{\partial t} + \frac{\partial \mathrm{R}(\mathrm{v})}{\partial x} = 0 \tag{6}$$

where $\mathrm{R}(\mathrm{v}) = \mathrm{v}^2 - \mathrm{v}_{\max}\mathrm{v}$. Variable change $\mathrm{v} = \mathrm{v} - \frac{\mathrm{v}_{\max}}{2}$ transforms Equation 6 into its final form:

$$\frac{\partial \mathrm{v}}{\partial t} + \frac{\partial \mathrm{v}^2}{\partial x} = 0$$

on the domain $(\mathrm{x},\mathrm{t}) \in \langle \mathrm{a},\mathrm{b}\rangle \times \langle 0,\mathrm{T}\rangle$. The initial and boundary conditions are too long to be presented here and can be found for example in [7]. For practical implementation, the LWR-v PDE is discretized using a Godunov numerical scheme [8]. By application of Godunov scheme, I obtain CTM-v. Let $\mathrm{n} \in \mathrm{N}$ be number of time steps of length $\delta t = \frac{\mathrm{T}}{\mathrm{N}}$, $\mathrm{M} \in \mathrm{N}$ number

of space cells of length $\delta x = \frac{b-a}{M}$. Then v_i^n is called discrete value of v at time step n in cell i ($0 \le n \le N$ and $0 \le i \le M$). At each time step v_i^{n+1} is computed from the previous time step. This is done by formula:

$$v_i^{n+1} = v_i^n - \frac{\Delta t}{\Delta x}\left(g(v_i^n, v_{i+1}^n) - g(v_{i-1}^n, v_i^n)\right) \quad (7)$$

where the numerical flow function g is:

$$g(v_1, v_2)\begin{cases} R(v_2) & if\ v_1 \le v_2 \le v_c \\ R(v_c) & if\ v_1 \le v_c \le v_2 \\ R(v_1) & if\ v_c \le v_1 \le v_2 \\ max(R(v_1), R(v_2)) & if\ v_1 \le v_2 \le v_c \end{cases}$$

where $v_c = \frac{v_{max}}{2}$, $R(v) = v^2 - v_{max}v$ and v_{max} is maximum speed allowed in the model. Note that Δt and Δx must satisfy Courant-Friedrichs-Lewy (CFL) condition:

$$\left|\frac{\Delta t}{\Delta x} v_{max}\right| \le 1$$

For implementation of boundary conditions, I use ghost cells placed at each side of the domain defined by the boundary conditions to be satisfied.

3 Algorithm

Ensemble Kalman filter based algorithm is presented in this part. Parts of this approach were presented in [9] and [10]. Once again, this algorithm utilizes both historical and current data from both my data sources. Main difference between this algorithm and the other ones is, that this one predicts the speed on the entire motorway and not only parts of it. Also, this algorithm is not only based on the historical data but also on a physical model. As a standalone, physical model (in my case CTM-v) would be unable, due to the missing data regarding its boundary conditions, cope with this task. However, thanks to the smart combination of historical data and physical model it is possible to utilize it in the prediction. Moreover, thanks to the integration of CTM-v into Ensemble Kalman filter, efficient assimilation of current situation data is guaranteed. Ensemble Kalman filter was chosen, because the problem of the traffic modelling is generally considered to be non-linear and not suitable for general Kalman filter. Due to the assimilation step, this algorithm is more useful for relatively short predictions until next traffic state data are available (5 minutes in my case because of ASIMs). General progress of this algorithm can be summarized into following points (note that steps 3 and 4 are performed repetitively while the prediction is being performed and current data supplied):

1. Learn the boundary conditions from the historical data.
2. Initiate an ensemble.
3. Perform an ensemble of predictions based on the learned physical model.
4. Assimilate current state data by EnKF.

Let us discuss each of these steps in detail. In the first step, learning of the boundary conditions is performed. It can be likened to the inverse modelling task of finding a set of boundary conditions B, such that

$$\mathbf{B} = \underset{b}{argmin}\ |\mathbf{m}(\mathbf{x\_0}, \mathbf{b}) - \mathbf{x}|, \quad (8)$$

where b are possible boundary conditions, x are available data and function m represents my physical model (i.e. I am minimizing the difference between the physical model and data given the boundary conditions). Given that I need to define boundary condition for each ramp on the motorway and that this condition should change in time, this optimization is not a simple one. It can be expected, that minimizing Equation 8 would be very difficult due to the high number of local minimums. Solution of this problem was found in the field of evolutionary computations and is named Differential evolution. This optimization technique is well proven to handle such difficult optimization problems as for example is proven in [9]. Optimized variable b is a matrix where rows represent individual ramps and columns its progress in time (this forms my specimen). Crossbreeding between specimen is then performed between rows of the b. In an optimal situation, model should be optimized for a week-long period, as the week is larges strictly periodical time interval in traffic. These specimens are then inserted into a CTM-v model and their results are compared to the original data producing the measure of quality. Result of this optimization is utilized in the step 3.

The second step is easy and straightforward. $\mathbf{N}$ members of ensemble are initialized by using free flow speed $\mathbf{v_{ff}}$ randomized by variance $\boldsymbol{\sigma^2}$ obtained from the historical data for each cell

$$\mathbf{v_i} = \mathbf{v_{ff}} + \mathbf{N}(\mathbf{0}, \boldsymbol{\sigma^2}),$$

where $\mathbf{v_i}$ is initial speed in the cell.

The next phase is a prediction phase. This phase is based on CTM-v and is realized by a modification of Equation 7.

$$\mathbf{v_i^{n+1}} = \mathbf{v_i^n} - \frac{\mathbf{\Delta t}}{\mathbf{\Delta x}}(\mathbf{g}(\mathbf{v_i^n}, \mathbf{v_{i+1}^n}) - \mathbf{g}(\mathbf{v_{i-1}^n}, \mathbf{v_i^n}) \pm \mathbf{B_i^n}),$$

where $\mathbf{B_i^n}$ is control variable governing the boundary conditions and learned in previous step. Addition or deduction depend whether it is in-ramp (-) or off-ramp (+). Length of each cell $\mathbf{x}$ and time step $\mathbf{t}$ must be set to adhere to Courant-Friedrichs-Lewy conditions. This means that if I want to have a suitably small cell, each model step must be reasonably short, and it is necessary to advance the model many times each prediction phase by calculating Equation 7 (for example, 5 minute prediction requires 300 prediction steps without any further data). This results in problem of finding balance

between detail and accuracy and will be topic for the future research. Because this algorithm is using EnKF, this prediction is not only done once but for each **N** members of the ensemble. Resulting prediction from the entire ensemble can be obtained by calculating an interval of confidence for mean speed of the ensemble (as usually ensemble has more than a few members).

The last step of my algorithm is current state data assimilation. This step is performed once per minute or per 5 minutes depending whether ASIM sensor data are used. It is also quite straightforward as this step consist only of calculating ensemble prediction covariance $\mathbf{P_t}$, calculating Kalman gain K_k and updating the ensemble by combining the new measurements with current ensemble. Measurements covariance **R** is calculated from historical data. Measurements used for this step come mostly from the FCD (TNC segments are divided into cells with the same measured value) and also can come from ASIM, where they are related to the cell with the gate. This provides us with more accurate measurement but also lengthens my prediction step. The entire algorithm can be summarized into following pseudo code.

EKF predict{**X**, **Xc**, **N**, **R**, **evol**}

1. **X**: historical FCD and ASIM values
2. **Xc**: current speed time series
3. **N**: ensemble size
4. **R**: measurements covariance
5. **evol**: vector with settings for DE
6. **B**=LearnDE(**X**, **evol**);
   \ *Learning of boundary conditions by DE*
7. **En**=InitializeEnsemble(**N**, **X**);
   \ *Ensemble initialization*
8. While {$\mathbf{x_c}$}
9. **EnsemP**=PredictEnsemble(**Ensem**, **B**);
   Prediction step
10. **Ensem**=Assimilate(**Xc**, **EnsemP**, **R**);
    Assimilation of current measurements
11. EndWhile

4 Experimental results

Experiments with this algorithm were performed on data from D1 (roughly from Humpolec to Brno) coming from 21.4.2014. Unfortunately, quantity of the data is insufficient to perform thorough testing and evaluation, however I do not possess larger data set at the moment. These results should be therefore considered as demonstration of usability. This problem would be solved soon with access to new datasets.

Experiment was performed only on FCD data, however, I kept 5-minute prediction step. I used 10 hours from 6:00 to 16:00 for learning and measurement covariance estimate. I used 50 generations of 75 individuals and set crossbreeding threshold to 0.7 and mutation constant to 1.2. After learning the model, I performed 20 prediction and assimilation steps. Spatial discretization for CTM-v model was chosen to be 100 m and temporal 1 second (by the CFL conditions). Results from prediction steps 6, 12 and 20 can be seen in Figures 2,3 and 4. These Figures show true traffic speed on entire section of the D1 motorway in the given step and prediction confidence interval.

RMSEs of these predictions are 8.65, 10.10 and 12.31 respectively. While these numbers may not seem to too high, few glances at the Figures 2, 3 and 4 show that results are roughly capable of capturing the magnitude of speed but rarely the trend. This fact seriously degrades value of the results. However, these problems are probably caused by small training set and may be solved by introduction of the new and larger data sets.

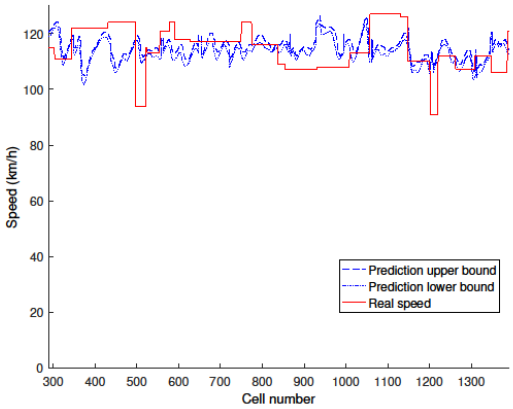

Figure 2 Traffic speed on segment of D1 after prediction step 6

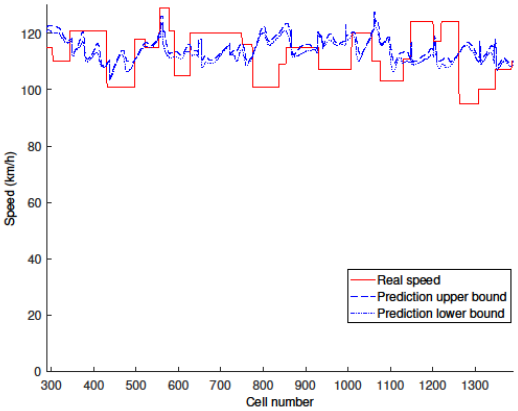

Figure 3. Traffic speed on segment of D1 after prediction step 12

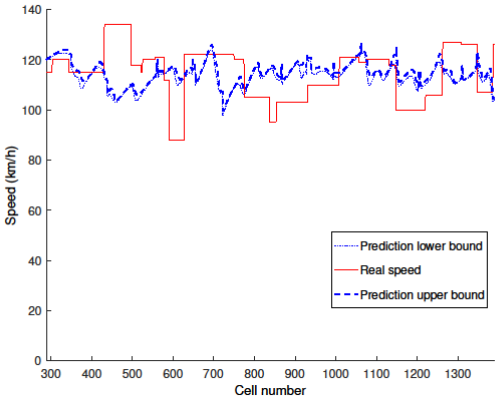

Figure 4. Traffic speed on segment of D1 after prediction step 20

For the comparison to the contemporary works, I have chosen articles written by Coric et. al. [11], Jiang et. al. [12], Castillo et. al. [13] and Xie et. al [14]. Coric et. al.

[11] used Mean Absolute Error (MAE) which are generally smaller than RMSE for equivalent data. These works were chosen because they present similar solutions and also have comparable results. Resulting errors of their algorithms are shown in Table 1.

Table1. Comparison of errors (RMSE or MAE) of contemporary works

Author	Jiang et. al. [12]	Castillo et. al. [13]	Coric et. al. [11]	Xie et. Al [14]
Average RMSE	9	8.7	6.2 (MAE)	9.1

From the table, it is evident that my algorithms have similar error results, even with small data sets. HMM algorithm even performs better than the other ones. Also, advantage of my algorithms is, that they were tested on real data with all of their problems and uncertainties. This is for example not true in case of Jiang et. al. [35] who used artificial data from SUMO traffic simulator.

4 Conclusion

From the scientific perspective, this work contains utilizes some interesting and original solutions. While proposed algorithms are generally based on existing methods, some subtasks required development of specific approaches. The most important innovation of EnKF algorithm is utilization of differential evolution for inverse modelling of boundary conditions. Perhaps more important achievement of this work is from the perspective of its application. Experimental results of all proposed algorithms indicate their usefulness in real intelligent traffic systems. From the perspective of future work, developed algorithms can still utilize some improvements to overcome their minor issues (other than enlarging the learning data sets). When trained on the suitably large data set, EnKF algorithm will be improved by thorough calibration. The purpose of this calibration is to find the best spatial and temporal discretization and also the optimal prediction window. The algorithm will also be extended for modelling of other traffic quantities, namely traffic density and intensity. Combination of these attributes can be utilized in our models to provide higher level of prediction accuracy. This combination could improve the prediction because of known interaction of these attributes, which simplifies identification of different traffic situations and reduces uncertainty.

5 Acknowledgment

This work was supported by The Ministry of Education, Youth and Sports from the National Programme of Sustainability (NPU II) project „IT4Innovations excellence in science - LQ1602“.

References

1. T.-Q. Tang, L. Caccetta, Y.-H. Wu, H.-J Huan, and X.-B. Yang. A macro model for traffic flow on road networks with varying road conditions. *Journal of Advanced transportation* **48**, pp. 304–317 (2014).
2. T. Q. Tang, J. G. Li, H. J. Huang, and X. B. Yang. A car-following model with real-time road conditions and numerical tests. *Measurement* **48** (2014).
3. G. Costesequea and J. P. Lebacque. A variational formulation for higher order macroscopic traffic flow models: Numerical investigation. *Transportation Research Part B: Methodological* **70** (2014).
4. S. C. Calvert, T. Henk, M. Snelder, and S. P. Hoogendoorn. Application of advanced sampling for efficient probabilistic traffic modelling. *Transportation Research Part C: Emerging Technologies* **49** (2014).
5. M.-L. Huang. Intersection traffic flow forecasting based on v-gsvr with a new hybrid evolutionaryalgorithm. *Neurocomputing* **147**, pp. 343–349 (2015).
6. K. Price. An introduction to differential evolution. *New Ideas in Optimization*, pp. 79–108 (1999).
7. D. Work, O. Tossavainen, S. Blandin, A. Bayen, T. Iwuchukwu, and K. Tracton. An ensemble kalman filtering approach to highway traffic estimation using gps enabled mobile devices. *In Proceedings of the 47th IEEE Conference on Decision and Control* (2008).
8. J. P. Lebacque. The godunov scheme and what it means for first order traffic flow models. In *Proceedings of 13th International Symposium on Transportation and Traffic Theory* (1996).
9. P. Dubec, J. Plucar, and L. Rapant. Use of the bio-inspired algorithms to find global minimum in force directed layout algorithms. *In Proceedings of the 6th International Conference on Multimedia Communications, Services and Security*, *Communications in Computer and Information Science* **368**, pp. 194–203. Springer Verlag (2013).
10. L. Rapant. Traffic flow modeling based on sparse data. *In Proceedings of WOFEX 2014*, pp. 526–531. VSB-TUO (2014).
11. S. Vucetic V. Coric, N. Djuric. Traffic state estimation from aggregated measurements with signal reconstruction techniques. *Transportation Research Record: Journal of the Transportation Research Board*, **2315** (2012).
12. B. Jiang and Y. Fei. Traffic and vehicle speed prediction with neural network and hidden markov model in vehicular networks. *In Proceedings of 2015 IEEE Intelligent Vehicles Symposium* (2015).
13. E. Castillo. Stochastic demand dynamic traffic models using generalized beta-gaussian bayesian networks. *IEEE Transactions on Intelligent Transportation Systems* **13** (2012).

14. Y. Xie, Y. Zhang, and Z. Ye. Short-term traffic volume forecasting using kalman filter with discrete wavelet decomposition. *Computer-Aided Civil and Infrastructure Engineering* **22**, pp. 326–334 (2007).